

Activity Planning and GraphPlan

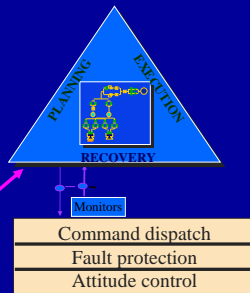
Slides draw upon material from:
Prof. Maria Fox

Brian C. Williams
16.410-13
Session 12

Autonomous Agents

Self-commanding
Self-diagnosing
Self-repairing

Commanded at:
• Mission level
• Engineering level



Mission Goal Scenario

Reading and Assignment for Planning Lectures

- Graph-based Planning
AIMA Chapter 11
- AIMA = "Artificial Intelligence: A Modern Approach," by Russell and Norvig.
- Assignment
 - No problem set out this week, due to mid-term.
 - Please take the practice mid-term.

Outline

- Example: MER Mission-Planning
- The Operator-based Planning Problem
- Plan Graphs
 - Solution to A Graph Plan Problem
 - Plan Graph Construction

Mars Exploration Rovers – Jan. 2004 - ?

Mini-TES
Pancam
Navcam

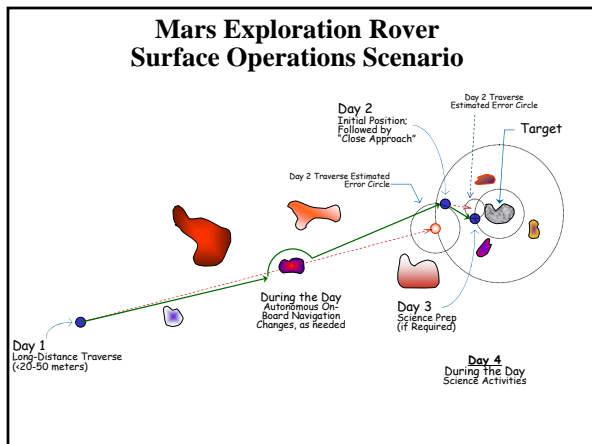


Mossbauer spectrometer
APXS
Rock Abrasion Tool
Microscopic Imager

Mission Objectives:

- Learn about ancient water and climate on Mars.
- For each rover, analyze a total of 6-12 targets
 - Targets = natural rocks, abraded rocks, and soil
- Drive 200-1000 meters per rover
- Take 1-3 panoramas both with Pancam and mini-TES
- Take 5-15 daytime and 1-3 nighttime sky observations with mini-TES

Courtesy of Kanna Rajan, NASA Ames. Used with permission.



Courtesy of Kanna Rajan, NASA Ames. Used with permission.

Outline

- Example: MER Mission Planning
- The Operator-based Planning Problem
- Plan Graphs

Planning

Find:
program of actions that achieves the objective

Planning

Find:
program of actions that achieves the objective

↓
partially-ordered set
typically unconditional

↓
goals

Paradigms

Classical planning

(STRIPS, operator-based, first-principles)
"generative"

Hierarchical Task Network planning

"practical" planning

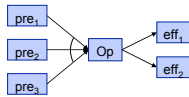
MDP & POMDP planning

planning under uncertainty

The Classical Representation

Initial Conditions: P_1 P_2 P_3 P_4

Operators:



Goals:

$Goal_1$ $Goal_2$ $Goal_3$

Simple Spacecraft Problem

Observation-1
target
instruments

pointing

Observation-2

calibrated

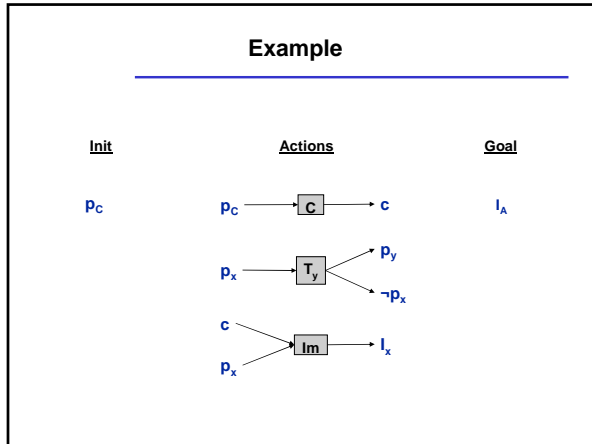
Observation-3

Observation-4

...



States: Target Pointed To, Camera Calibrated?, Has Image?
Operations: Turn, Take Image, and Calibrate.



Operators in STRIPS Representation

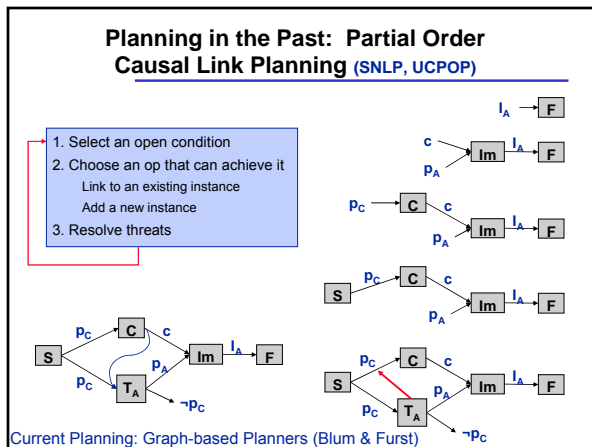
TakeImage (?target, ?instr):
 Pre: Status(?instr, Calibrated), Pointing(?target)
 Eff: Image(?target)

Calibrate (?instrument):
 Pre: Status(?instr, On), Calibration-Target(?target), Pointing(?target)
 Eff: \neg Status(?instr, On), Status(?instr, Calibrated)

Turn (?target):
 Pre: Pointing(?direction), ?direction \neq ?target
 Eff: \neg Pointing(?direction), Pointing(?target)

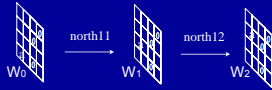
Often begin with “?” to denote a parameter, as in ?var.

Based on slides by Dave Smith, NASA Ames



Operator-based Planning Problem

- **Input**
 - Set of world states
 - Action operators
 - Fn: world-state → world-state
 - Initial state of world
 - Goal
 - A partial state (set of world states)
- **Output**
 - Sequence of actions that is
 - Complete: Achieve goals
 - Consistent: No negative side-effects



20

Example Problem: Dinner Date

Initial Conditions: (and (cleanHands) (quiet))

Goal: (and (noGarbage) (dinner) (present))

Actions:

- (:operator **carry** :precondition :effect (and (noGarbage) (not (cleanHands))))
- (:operator **dolly** :precondition :effect (and (noGarbage) (not (quiet))))
- (:operator **cook** :precondition (cleanHands) :effect (dinner))
- (:operator **wrap** :precondition (quiet) :effect (present))
- + noops

Solution: (Cook, Wrap, Carry)

Representing States

- State
 - A consistent conjunction of propositions (positive literals).
 - E.g., (and (cleanhands) (quiet) (dinner) (present) (noGarbage))
 - All unspecified propositions are false
- Initial State
 - Problem state at time $i = 0$
 - E.g., (and (cleanHands) (quiet))
- Goal State
 - A partial state
 - E.g., (and (noGarbage) (dinner) (present))
 - The planner must generate an action sequence that places the system in a final state that satisfies the goal conjunction.

22

Representing Operators

```
(:operator carry
  :precondition
  :effect (:and (noGarbage)
               (not (cleanHands))))
```

Note: STRIPS doesn't allow derived effects: you must be complete!

Preconditions: Propositions that must be true to apply the operator.

- A conjunction of propositions (no negated propositions).

Effects: Propositions that the operator changes, given the preconditions are satisfied.

- A conjunction of propositions (called adds) and their negation (called deletes).

23

Operator Execution Semantics

If all propositions of :precondition appear in state i,

Then create state i+1 from i, by

- adding to i, all "add" propositions in :effects,
- removing from i, all "delete" propositions in :effects.

```
(:operator cook :precondition (cleanHands)
  :effect (dinner))
```

```
(cleanHands)
(quiet)      → cook → (cleanHands)
                    (quiet)
                    (dinner)
```

24

Operator Execution Semantics

If all propositions of :precondition appear in state i,

Then create state i+1 from i, by

- adding to i, all "add" propositions in :effects,
- removing from i, all "delete" propositions in :effects.

```
(:operator dolly :precondition
  :effect (and (noGarbage) (not (quiet))))
```

```
(cleanHands)
(quiet)      → dolly → (cleanHands)
                    (noGarbage)
```

25

(Parameterized) Operator Schemata

- Instead of defining many operator instances: **pickup-A** and **pickup-B** and ...
- Define a schema:

```
(:operator pick-up
  :parameters ((block ob1))
  :precondition (and (clear ob1)
                    (on-table ob1)
                    (arm-empty))
  :effect (and (not (clear ob1))
              (not (on-table ob1))
              (not (arm-empty))
              (holding ob1)))
```

Often begin with "?" to denote a parameter, as in ?var.

What World Assumptions are Implied by the STRIPS Representation?

```
(:operator pick-up
  :parameters ((block ob1))
  :precondition (and (clear ob1)
                    (on-table ob1)
                    (arm-empty))
  :effect (and (not (clear ob1))
              (not (on-table ob1))
              (not (arm-empty))
              (holding ob1)))
```

- Atomic time.
- Agent is omniscient (no sensing necessary).
- Agent is sole cause of change.
- Actions have deterministic effects.
- No indirect effects.

⇒ STRIPS Assumptions

28

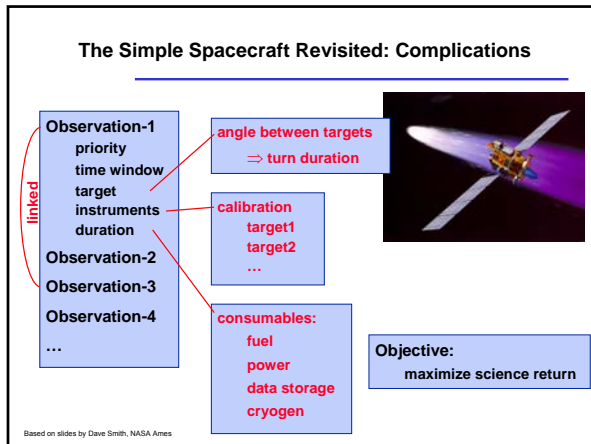
The Simple Spacecraft Revisited: Complications

Observation-1
priority
time window
target
instruments
duration
Observation-2
Observation-3
Observation-4
...

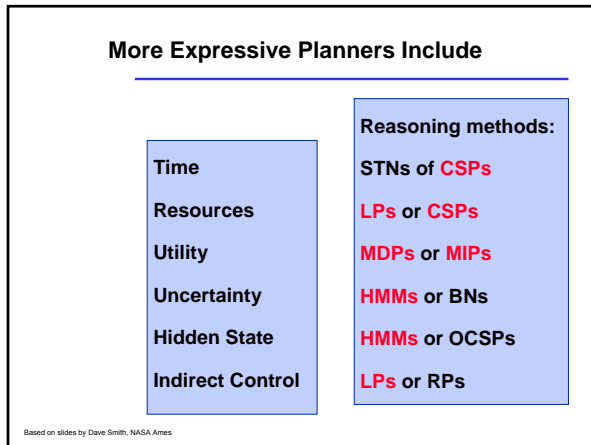


Objective:
maximize science return

Based on slides by Dave Smith, NASA Ames



Courtesy of NASA.



Outline

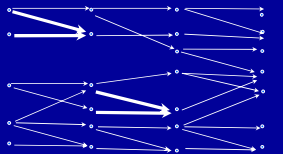
- Example: MER Mission Planning
- The Operator-based Planning Problem
- Plan Graphs
 - Solution to A Graph Plan Problem
 - Plan Graph Construction

Graph Plan

- GraphPlan was developed in 1995 by Avrim Blum and Merrick Furst, at CMU.
- The **Plan Graph encoding** of state space has been a **key to scaling up** to realistic problems.
- The GraphPlan approach has been **extended** to reason with **temporally extended actions, metric and non-atomic preconditions and effects**.
- Plan Graph representation used for:
 - **Relaxed planning** as an **admissible heuristic**.
 - An **encoding** method for formulating planning **as a CSP**.

Approach: Graph Plan

1. Constructs compact **constraint encoding** of state space from operators and initial state, **which prunes many invalid plans** – *Plan Graph*.
2. **Generates plan by searching** for a consistent subgraph that achieves the goals.



Proposition
Init State Action
Time 1 Proposition
Time 1 Action
Time 2

Outline

- Example: MER Mission Planning
- The Operator-based Planning Problem
- Plan Graphs
 - **Solution to A Graph Plan Problem**
 - Plan Graph Construction

Visualizing Actions in a Plan Graph

(:operator **cook** :precondition (cleanHands)
:effect (dinner))



(:operator **carry** :precondition
:effect (:and (noGarbage) (not (cleanHands))))



36

Visualizing Actions in a Plan Graph

• Persistence actions (Noops)

- Every literal has a no-op action, which maintains it from time i to $i+1$.

(:operator **noop-P** :precondition (P) :effect (P))



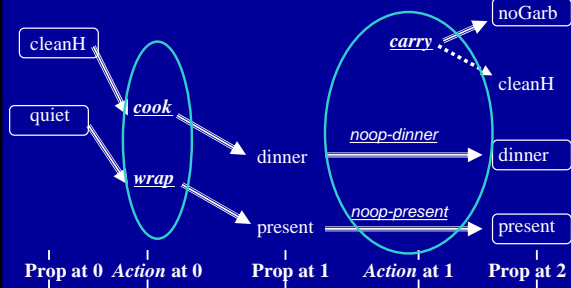
In Blum & Furst: (& lecture)
AIMA:

Only persist positive literals.
Persists negative literals as well.
either approach okay for PSets.

37

A Plan in GraphPlan <Actions[i] >

- Sets of concurrent actions are performed at each time [i]
 - Concurrent actions can be interleaved in any order.
 - ⇒ If actions **a** and **b** occur at time i , then it must be valid to perform either **a** followed by **b**, OR **b** followed by **a**.



A Complete Consistent Plan

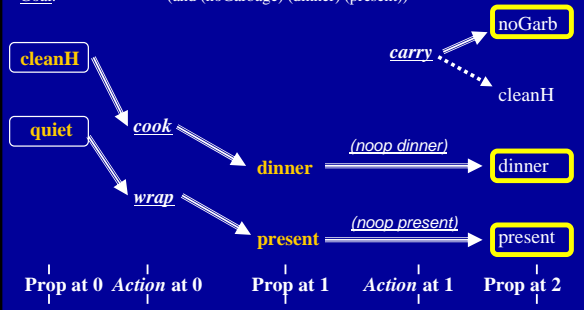
Given that the initial state holds at time 0,
a plan is a solution iff:

- Complete:
 - The **goal propositions** all hold in the **final state**.
 - The **preconditions** of every operator at time i , is **satisfied by a proposition** at time i .
- Consistent:

Example of a Complete Consistent Plan

Initial Conditions: (and (cleanHands) (quiet))

Goal: (and (noGarbage) (dinner) (present))



A Complete Consistent Plan

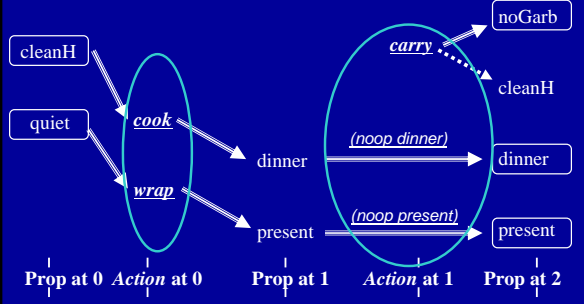
Given that the initial state holds at time 0,
a plan is a solution iff:

- Complete:
 - The goal propositions all hold in the final state.
 - The preconditions of every operator at time i , is satisfied by a proposition at time i .
- Consistent:
 - The **operators** at any time i can be **executed in any order**, **without one of these operators undoing:**
 - the **preconditions** of another operator at time i .
 - the **effects** of another operator at time i .

Example of a Complete Consistent Plan

Initial Conditions: (and (cleanHands) (quiet))

Goal: (and (noGarbage) (dinner) (present))

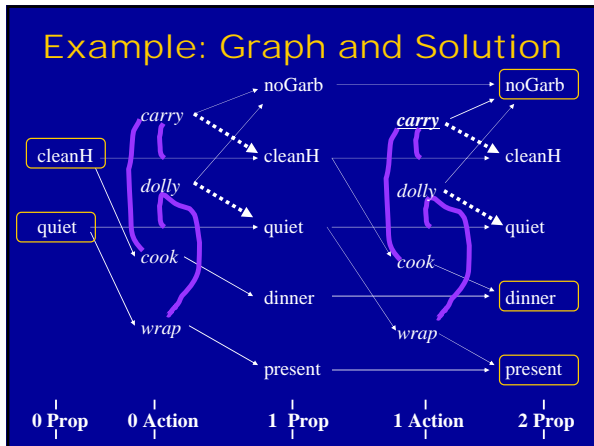


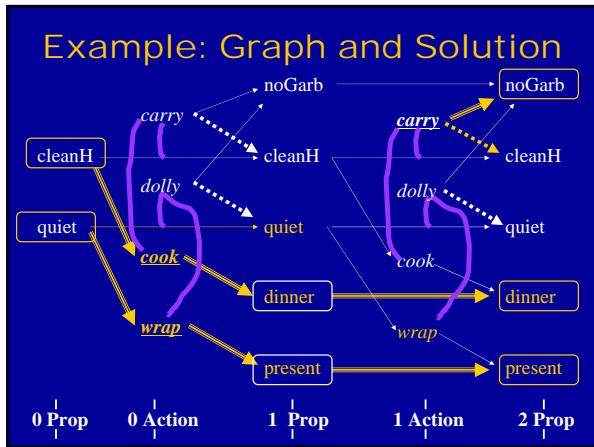
GraphPlan Algorithm

- Phase 1 – Plan Graph Expansion
 - Graph includes, as a subset, all plans that are complete and consistent, while pre-pruning many infeasible plans.
- Phase 2 - Solution Extraction
 - Graph is a kind of constraint satisfaction problem (CSP).
 - Extraction selects actions to perform at each time point, by assigning CSP variables and testing consistency.

Outline

- Example: MER Mission Planning
- The Operator-based Planning Problem
- Plan Graphs
 - Solution to A Graph Plan Problem
 - Plan Graph Construction





Graph Properties

A Plan graph

- compactly encodes the space of consistent plans,
- while pruning . . .
 1. partial states and actions at each time i that are **not reachable** from the **initial state**.
 2. pairs of propositions and actions that are **mutually inconsistent** at time i .
 3. plans that **cannot reach the goals**.

Graph Properties

- Plan graphs are constructed in **polynomial time** and are of **polynomial in size**.
- The plan graph **does not eliminate** all infeasible plans.
- Plan generation still **requires *focused* search**.

Constructing the plan graph... (Reachability)

- Initial proposition layer
 - Contains propositions that hold in the initial state.

Example: Initial State, Layer 1

cleanH

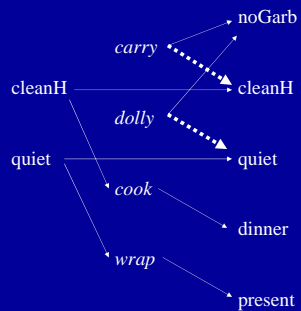
quiet

0 Prop 0 Action 1 Prop 1 Action 2 Prop

Constructing the plan graph... (Reachability)

- Initial proposition layer
 - Contains propositions in initial state
- Action layer *i*
 - If all of action's preconditions are consistent in proposition layer *i*
 - Then add action to layer *i*
- Proposition layer *i+1*
 - For each action at layer *i*
 - Add all its effects at layer *i+1*

Example: Add Actions and Effects



0 Prop 0 Action 1 Prop 1 Action 2 Prop

Constructing the planning graph...(Reachability)

- Initial proposition layer
 - Write down just the initial conditions
- Action layer *i*
 - If all action's preconditions appear consistent in proposition layer *i*
 - Then add action to layer *i*
- Proposition layer *i+1*
 - For each action at layer *i*
 - Add all its effects at layer *i+1*
- Repeat adding layers until all goal propositions appear

Round 1: Stop at Proposition Layer 1?



Do all goal propositions appear?

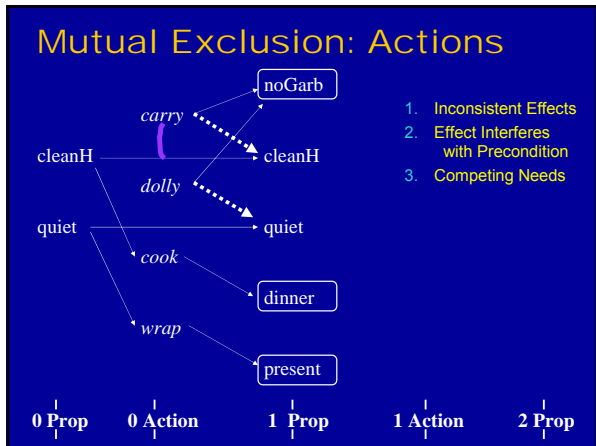
Goal: (and (noGarbage) (dinner) (present))

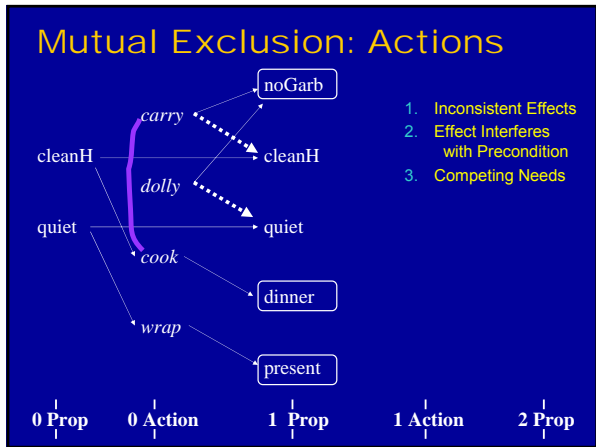
Constructing the plan graph... (Consistency)

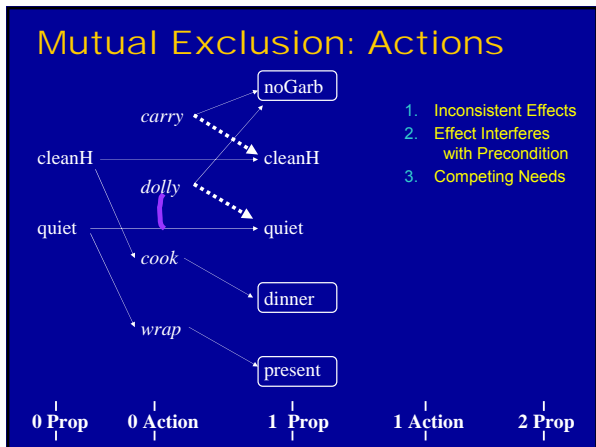
- Initial proposition layer
 - Write down just the initial conditions
- Action layer i
 - If action's preconditions appear **consistent** in $i-1$ (**non-mutex**)
 - Then add action to layer i
- Proposition layer $i+1$
 - For each action at layer i
 - Add all its effects at layer $i+1$
- Identify mutual exclusions
 - Actions in layer i
 - Propositions in layer $i + 1$
- Repeat until all goal propositions appear **non-mutex**

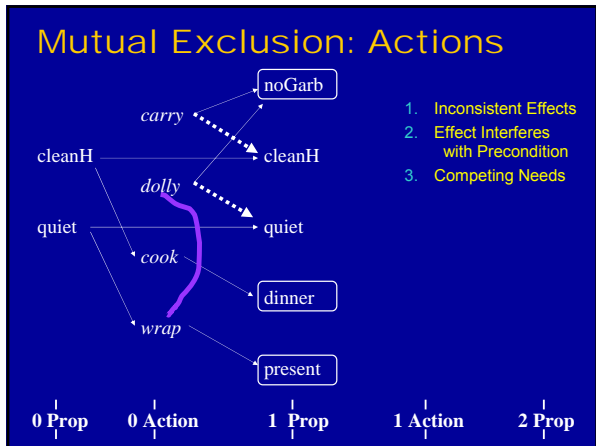
Mutual Exclusion: Actions

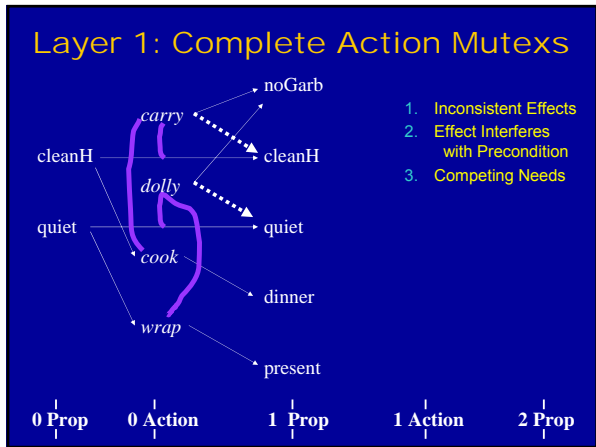
- Actions A,B are **mutually exclusive at level i** if **no valid plan** could possibly **contain both** at i :
 - They have **inconsistent effects**.
 - A deletes B's effects,
 - **Effects interfere** with preconditions.
 - A deletes B's preconditions, or
 - Vice versa or
 - They **compete for needs**.
 - A and B have **inconsistent preconditions**

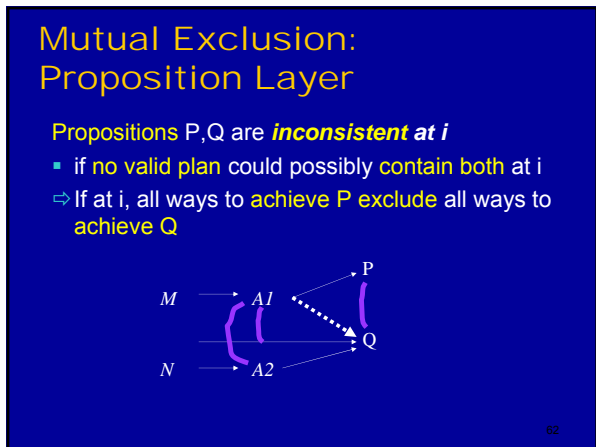




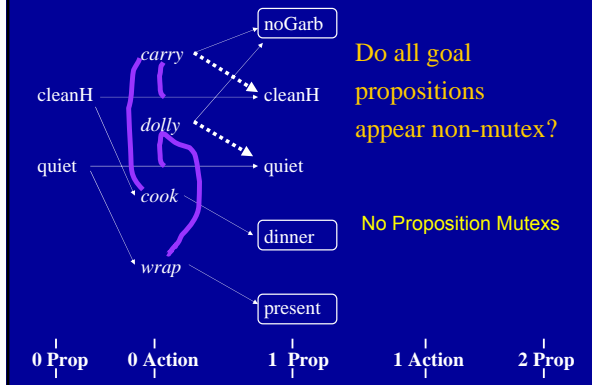




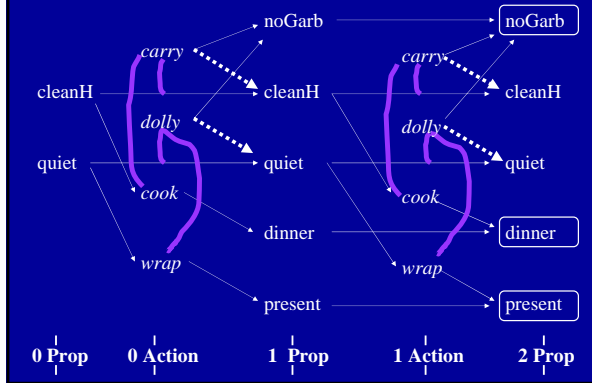




Layer 1: Add Proposition Mutexs



Round 2: Extending The Plan Graph



Outline

- Example: MER Mission Planning
- The Operator-based Planning Problem
- Plan Graphs
 - Solution to A Graph Plan Problem
 - Plan Graph Construction
